

Partial Searchlight Scheduling is Strongly PSPACE-complete

Giovanni Viglietta*

Abstract

The problem of searching a polygonal region for an unpredictably moving intruder by a set of stationary guards, each carrying an orientable laser, is known as the Searchlight Scheduling Problem. Determining the complexity of deciding if the entire area can be searched is a long-standing open problem. Recently, the author introduced the Partial Searchlight Scheduling Problem, in which only a given subregion of the environment has to be searched, and proved that its 3-dimensional decision version is **PSPACE**-hard, even when restricted to orthogonal polyhedra.

Here we extend and refine this result, by proving that 2-dimensional Partial Searchlight Scheduling is strongly **PSPACE**-complete, both in general and restricted to orthogonal polygons in which the region to be searched is a rectangle.

1 Introduction

The *Searchlight Scheduling Problem* (SSP), first studied in [3], is a pursuit-evasion problem in which a polygon has to be searched for a moving intruder by a set of stationary guards. The intruder moves unpredictably and continuously with unbounded speed, and each guard carries an orientable *searchlight*, emanating a 1-dimensional ray that can be continuously rotated about the guard itself. The polygon's exterior cannot be traversed by the intruder, nor penetrated by searchlights. The intruder is caught whenever it is hit by a searchlight. Because the intruder's location is unknown until it is actually caught, each guard has to sway its searchlight according to a predefined *schedule*. If the guards always catch the intruder, regardless of its path, by following their schedules in concert, they are said to have a *search schedule*.

SSP is the problem of deciding if a given set of guards has a search schedule for a given polygon (with holes). The computational complexity of this decision problem has been only marginally addressed in [3], but has later gained more attention, until in [2] the space of all possible schedules has been shown to be discretizable and reducible to a finite graph, which can be explored exhaustively in order to find a search schedule, if one exists. Since the graph may have double exponential size, this technique easily places SSP

in **2-EXP**. Whether SSP is **NP**-hard or even in **NP** is left in [2] as an open problem.

More recently, in [4], the author introduced the *Partial Searchlight Scheduling Problem* (PSSP), in which the guards content themselves with searching a smaller subregion given as input. That is, a search schedule should only guarantee that the given *target region* is eventually cleared, either by catching the intruder or by confining it outside. A 3-dimensional variation of PSSP is studied in [4], in which the input polygonal environment is replaced by an orthogonal polyhedron, and the 1-dimensional rays become 2-dimensional half-planes, which rotate about their boundary lines. Such a problem is shown to be strongly **PSPACE**-hard.

In the present paper we take a further step along this line of research, by proving that 2-dimensional PSSP is strongly **PSPACE**-complete, both in general and restricted to orthogonal polygons in which the region to be searched is a rectangle.

2 Asynchronous NCL machines

Our reduction is based on a model of computation described in [1] and [4], called *Nondeterministic Constraint Logic* (NCL).

An *asynchronous NCL machine* is a 3-regular graph, each of whose vertices is either an *AND vertex* or an *OR vertex*. Of the three edges incident to an AND vertex, one is called its *output edge*, and the other two are its *input edges*. Each edge of an asynchronous NCL machine can be oriented toward either one of its incident vertices (or none), and a configuration of edge orientations is *legal* if

- for each AND vertex, either its output edge is directed inward, or both its input edges are directed inward;
- for each OR vertex, at least one of its three incident edges is directed inward.

Accordingly, a *legal move* consists in the reversal of an edge's orientation that preserves legality of the configuration. Moves can occur at any time independently (i.e., *asynchronously*), and each reversal of an edge can take an arbitrarily long but finite time, during which that edge is not oriented toward any vertex.

Given an asynchronous NCL machine with two *distinguished edges* e_a and e_b , and a *target orientation*

*Department of Computer Science, University of Pisa, viglietta@gmail.com

for each, we call EE-ANCL the problem of deciding if there exist legal configurations A and B such that e_a has its target orientation in A , e_b has its target orientation in B , and there is an asynchronous sequence of legal moves from A to B . EE-ANCL is shown to be **PSPACE**-complete in [4], by a reduction from its *synchronous* version, thoroughly studied in [1].

3 PSPACE-completeness of PSSP

To prove that PSSP belongs to **PSPACE** we use the discretization technique of [2], and to prove that PSSP is **PSPACE**-hard we give a reduction from EE-ANCL.

Lemma 1 $PSSP \in PSPACE$.

Proof. As explained in [2], a technique known as *exact cell decomposition* allows to reduce the space of all possible schedules to a finite graph G . Each searchlight has a linear number of *critical angles*, which yield an overall partition of the polygon into a polynomial number of *cells*. Searchlights take turns moving, and can stop or change direction only at critical angles. Thus, a vertex of G encodes the *status* of each cell (either *contaminated* or *clear*) and the critical angle at which each searchlight is oriented.

As a consequence, G can be navigated nondeterministically by just storing one vertex at a time, which requires polynomial space. Notice that deciding if two vertices of G are adjacent can be done in polynomial time: An edge in G represents a move of a single searchlight between two consecutive critical angles, and the updated status of each cell can be easily evaluated. Indeed, cells' vertices are intersections of lines through input points, hence their coordinates can also be efficiently stored and handled as rational expressions involving the input coordinates.

Now, in order to verify that a path in G is a witness for SSP, one checks if the last vertex encodes a status in which every cell is clear. But the very same cell decomposition works also for PSSP: The analysis in [2] applies even if just a subregion of the polygon has to be searched, and a path in G is a witness for PSSP if and only if its last vertex encodes a status in which every cell that has a non-empty intersection with the target subregion is clear.

Since **PSPACE** = **NPSPACE**, due to Savitch's theorem, our claim follows. \square

For the **PSPACE**-hardness part, we first give a reduction in which the target region to be cleared is an orthogonal hexagon. Then, Section 4 will explain how we would have to modify our construction, should we insist on having a rectangular (hence convex) target region.

Lemma 2 $EE\text{-}ANCL \preceq_P PSSP$ restricted to orthogonal polygons.

Proof. We show how to transform a given asynchronous NCL machine G with two distinguished edges e_a and e_b into an instance of PSSP.

A rough sketch of our construction is presented in Figure 1. All the vertices of G are placed in a row (a), and are connected together by a network of thin *corridors* (b), turning at right angles, representing edges of G . Each *subsegment* of a corridor is a thin rectangle, containing a *subsegment guard* in the middle (not shown in Figure 1). Two subsegments from different corridors may indeed cross each other like in (c), but in such a way that the crossing point is far enough from the ends of the two subsegments and from the two subsegment guards (so that no subsegment guard can see all the way through another subsegment). All the vertices of G and all the *joints* between consecutive subsegments (i.e., the turning points of each corridor) are connected via extremely thin *pipes* (d) to the upper area (e), which contains the target region (shaded in Figure 1).

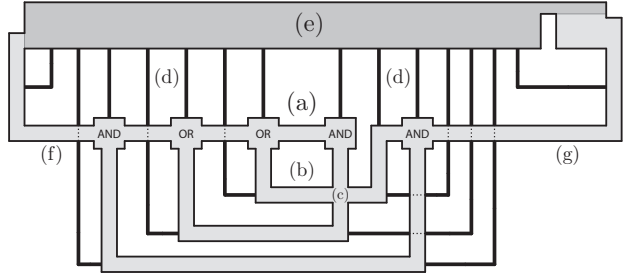


Figure 1: Construction overview.

Two corridors (f) and (g) also reach the upper area, and they correspond to the distinguished edges of G , e_a and e_b , respectively. That is, if $e_a = \{u, v\}$, and the target orientation of e_a is toward v , then the corridor corresponding to e_a connects vertex u in our construction to the upper area (e), rather than to v . The same holds for e_b . Indeed, observe that we may assume that e_a and e_b are reversed only once (respectively, on the first and last move) in a sequence of moves that solves EE-ANCL on G . As a consequence, contributions to vertex constraints given by distinguished edges oriented in their target direction may be ignored.

Each pipe turns at most once, and contains one *pipe guard* in the middle, lying on the boundary. Notice that straight pipes never intersect corridors, but some turning pipes do. Figure 2 shows a turning pipe, with its pipe guard (a) and an intersection with a corridor (b) (proportions are inaccurate). The *intersection guards* (c) separate the pipe from the corridor with their lasers (dotted lines in Figure 2), without “disconnecting” the pipe itself. Although a pipe narrows every time it crosses a corridor, its pipe guard

can always see all the way through it, because it is located in the middle. The small *nook* (d) is unclearable because no guard can see its bottom, hence it is a constant source of recontamination for the target region (e), unless the pipe guard is covering it with its laser. (Each straight pipe also has a similar nook.)

In our construction, corridor guards implement edge orientations in G : Whenever all the subsegment guards in a corridor connecting vertices u and v have their lasers oriented in the same “direction” from vertex u to vertex v , it means that the corresponding edge $\{u, v\}$ in G is oriented toward v .

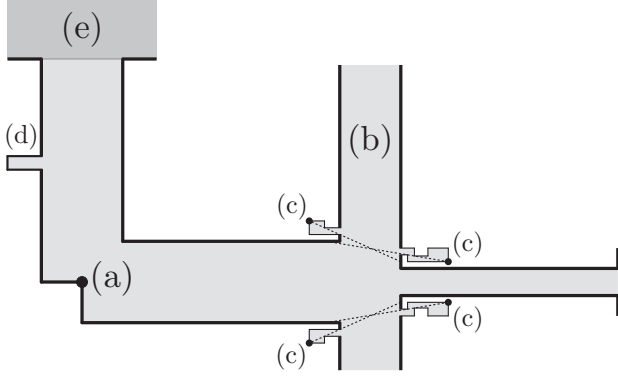


Figure 2: Intersection between a pipe and a corridor.

Figure 3 shows an OR vertex. The three subsegment guards from incoming corridors (a) can all “cap” pipe (b) with their lasers, and nook (c) guarantees that the pipe is recontaminated whenever all three guards turn their lasers away.

AND vertices are implemented as in Figure 4. The two subsegment guards (a) correspond to input edges, and are able to cap one pipe (e) each, whereas guard (c) can cover them both simultaneously. But that leaves pipe (d) uncovered, unless it is capped by guard (b), which belongs to the corridor corresponding to the output edge. Again, uncovered pipes are recontaminated by unclearable nooks (f).

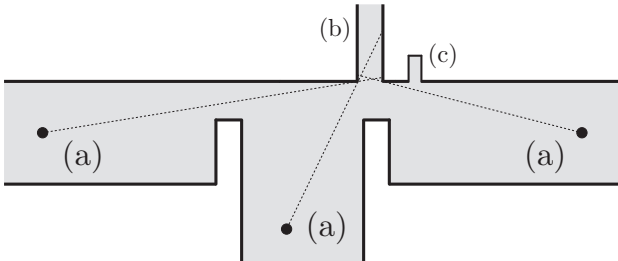


Figure 3: OR vertex.

Joints between consecutive subsegments of a corridor may be viewed as OR vertices with two inputs, shaped like in Figure 3, but without the corridor coming from the left.

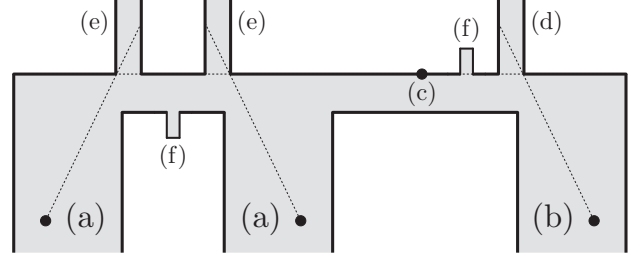


Figure 4: AND vertex.

Finally, Figure 5 shows the upper area of the construction, reached by the distinguished edges e_a and e_b (respectively, (a) and (b)), and by all the pipes (c). The guard in (d) can cap all the pipes, one at a time, and its purpose is to clear the left part of the target region, while the small rectangle (e) on the right will be cleared by the guard in (f). The two pipes (g) implement additional OR vertices with two inputs, and prevent (d) and (f) from acting, unless the respective distinguished edges are in their target orientations. Nook (h) will contaminate part of the target region, unless (d) is aiming down. Nooks (i) prevent area (e) from staying clear whenever guard (f) is not aiming up. The guard in (j) separates the two parts of the target region with its laser, so that they can be cleared in two different moments.

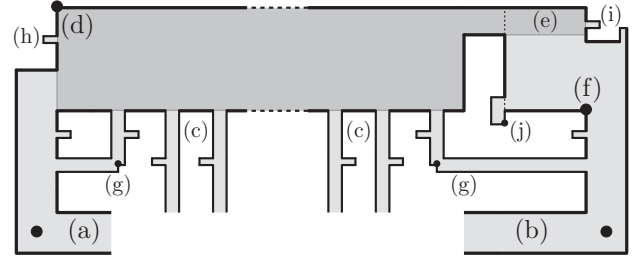


Figure 5: Target region.

Suppose G is a solvable instance of EE-ANCL. Then we can “mimic” the transition from configuration A to configuration B (see Section 2) by turning subsegment guards. Specifically, if edge $e = \{u, v\}$ in G changes its orientation from u to v , then all the subsegment guards in the corridor corresponding to e turn their lasers around, one at a time, starting from the guard closest to u . Before this process starts, each pipe has one end capped by some subsegment guard, and in particular pipe (g) on the left of Figure 5 is capped by the guard in (a). Hence, guard (d) is free to turn and cap all the pipes one by one, stopping for a moment to let each pipe’s internal guard clear the pipe itself (which now has both ends capped) and cover its nook (see Figure 2). As a result, the left part of the target region can be cleared by rotating (d) clockwise, from right to down. Then the subsegment

guards start rotating as explained above, until configuration B is reached. If done properly, this keeps all the pipes capped and clear, thus preventing the left part of the target region from being recontaminated. When B is reached, guard (f) can turn up to clear (e) and finally solve our PSSP instance.

Conversely, suppose that G is not solvable. Observe that rectangle (e) in Figure 5 has to be cleared by guard (f) as a last thing, because it will be recontaminated by nooks (i) as soon as (f) turns away. On the other hand, as soon as a pipe has both ends uncapped by external guards, some portion of the target region necessarily gets recontaminated by some nook, regardless of where the pipe guard is aiming its laser. But guard (d) can cap just one pipe at a time and, while it does so, nook (h) keeps some portion of the target region contaminated. Thus, the entire process must start from a configuration A in which all the pipes are simultaneously capped and guard (d) is free to turn right (i.e., e_a is in its target orientation), then proceed without ever uncapping any pipe (i.e., preserving legality), and finally reach a configuration B in which guard (f) is free to turn up (i.e., e_b is in its target orientation). By assumption this is impossible, hence our PSSP instance is unsolvable. \square

By putting together Lemma 1 and Lemma 2, we immediately obtain the following:

Theorem 3 *Both PSSP and its restriction to orthogonal polygons are strongly \mathbf{PSPACE} -complete.* \square

The term “strongly” is implied by the fact that all the vertex coordinates generated in the \mathbf{PSPACE} -hardness reduction of Lemma 2 are numbers with polynomially many digits (or can be made so through negligible adjustments).

4 Convexifying the target region

We can further improve our Theorem 3 by making the target region in Lemma 2 rectangular.

Our new target region has the same width as the previous one, and the height of rectangle (e) in Figure 5. In order for this to work, we have to make sure that some portion of the target region is “affected” by each contaminated pipe that is not capped by guard (d), no matter where *all* the pipe guards are oriented. To achieve this, we make pipes reach the upper area of our construction at increasing heights, from left to right, in a staircase-like fashion.

Assume we already placed pipe (a) in Figure 6, and we need to find the correct height at which it is safe to connect pipe (b). First we find the rightmost intersection (c) between a laser emanating from the pipe guard of (a) and the lower border of the target region. Then we set the height of pipe (b) so that it is capped by guard (d) when it aims slightly to the right of (c).

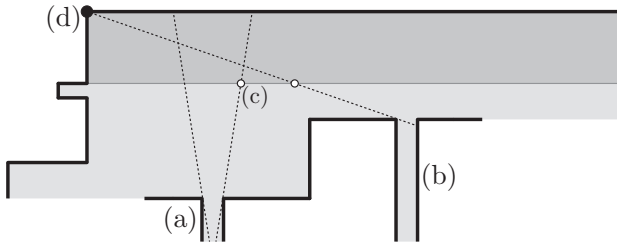


Figure 6: Rectangular target region.

This is always feasible, provided that pipes are thin enough, which is not an issue.

After we have set all pipes’ heights from left to right, the construction is complete and the proof of Lemma 2 can be repeated verbatim, yielding:

Theorem 4 *Both PSSP and its restriction to orthogonal polygons with rectangular target regions are strongly \mathbf{PSPACE} -complete.* \square

5 Further research

Observe that the target region constructed in [4] for 3-dimensional PSSP is an arbitrarily small ball centered at a given point. We could even modify PSSP by asking if any neighborhood of a given point is clearable at all (as opposed to a well-defined polygonal target region), and this problem would stay \mathbf{PSPACE} -hard for polyhedral environments. Our question is whether this holds true for 2-dimensional polygons, as well.

Similarly, we may investigate the complexity of PSSP on other restricted inputs, such as simply connected polygons, or target regions coinciding with the whole environment. The latter is in fact SSP, which has been mentioned in Section 1 as an interesting long-standing open problem. In [4], the author proved that the 3-dimensional version of SSP is \mathbf{NP} -hard, but determining the true complexity of either version still seems a deep problem and a possibly laborious task.

References

- [1] R. A. Hearn and E. D. Demaine. *PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation*. Theor. Comput. Sci. **343** (2005), 72–96, special issue “Game Theory Meets Theoretical Computer Science.”
- [2] K. J. Obermeyer, A. Ganguli, and F. Bullo. *A complete algorithm for searchlight scheduling*. Int. J. Comput. Geom. Ap. **21** (2011), 101–130.
- [3] K. Sugihara, I. Suzuki, and M. Yamashita. *The searchlight scheduling problem*. SIAM J. Comput. **19** (1990), 1024–1040.
- [4] G. Viglietta. *Searching polyhedra by rotating half-planes*. Preprint (2011), available at <http://arxiv.org/abs/1104.4137>.